MODIFICATION OF HILL CIPHER TECHNIQUE USING SELF REPETITIVE MATRIX (MODULAR ARITHMATIC) AND CORRELATION OF EIGEN VALUES OF MATRIX WITH THE EXPONENT N

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Electronics & Instrumentation Engineering

By

YELLAPU NAVEEN KUMAR

and

BIKKINA NARENDRA

Department of Electronics & Instrumentation Engineering

National Institute of Technology

Rourkela

2008

MODIFICATION OF HILL CIPHER TECHNIQUE USING SELF REPETITIVE MATRIX (MODULAR ARITHMATIC) AND CORRELATION OF EIGEN VALUES OF MATRIX WITH THE EXPONENT N

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Electronics & Instrumentation Engineering

By

YELLAPU NAVEEN KUMAR

and

BIKKINA NARENDRA

Department of Electronics & Instrumentation Engineering

National Institute of Technology

Rourkela

2008

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled, "Data encryption and decryption using Hill Cipher method and Self Repetitive Matrix" submitted by Sri Yellapu Naveen Kumar and Sri Bikkina Narendra in partial fulfillments for the requirements for the award of Bachelor of Technology Degree in Electronics & Instrumentation Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

Prof. G. S. Rath
Dept. of Electronics & Instrumentation Engg
National Institute of Technology
Rourkela - 769008

ACKNOWLEDGEMENT

We would like to articulate our deep gratitude to our project guide Prof. G.S.Rath who has always been our motivation for carrying out the project. It is our pleasure to refer Microsoft Word exclusive of which the compilation of this report would have been impossible. An assemblage of this nature could never have been attempted with out reference to and inspiration from the works of others whose details are mentioned in reference section. We acknowledge out indebtedness to all of them. Last but not the least, our sincere thanks to all of our friends who have patiently extended all sorts of help for accomplishing this undertaking.

YELLAPU NAVEEN KUMAR

BIKKINA NARENDRA

Contents

CHAPTER 1

GENERAL INTRODUCTION

Cryptography has a long and fascinating history. Over the centuries, an elaborate set of protocols and mechanisms has been created to deal with information security issues when the information is conveyed by physical documents. Often the objectives of information security cannot solely be achieved through mathematical algorithms and protocols alone, but require procedural techniques and abidance of laws to achieve the desired result. For example, privacy of letters is provided by sealed envelopes delivered by an accepted mail service. The physical security of the envelope is, for practical necessity, limited and so laws are enacted which make it a criminal offense to open mail for which one is not authorized. It is sometimes the case that security is achieved not through the information itself but through the physical document recording it. For example, paper currency requires special inks and material to prevent counterfeiting.

One of the fundamental tools used in information security is the signature. It is a building block for many other services such as non-repudiation, data origin authentication, identification, and witnessing, to mention a few. Having learned the basics in writing, an individual is taught how to produce a handwritten signature for the purpose of identification. At contract age the signature evolves to take on a very integral part of the person's identity. This signature is intended to be unique to the individual and serve as a means to identify, authorize, and validate. With electronic information the concept of a signature needs to be redressed; it cannot simply be something unique to the signer and independent of the information signed. Electronic replication of it is so simple that appending a signature to a document not signed by the originator of the signature is almost a triviality.

With the advent of the computer age, the mechanical encryption techniques were replaced with computer ciphers .They operated according to the same principles of substitution and transposition (where the order of letters or bits is altered). Again each cipher depended on

choosing a key, known only by the sender and the receiver which defined how a particular message would be. This meant that there still was a problem of getting the key to the receiver so that the message could be deciphered. This had to be done in advance, which was an expensive slow and risky process.

For yeas, this key distribution problem haunted code makers i.e. if you want to decipher a scrambled text you have to know the key in advance. But there was revolution in cryptography known as public key cryptography, which destroyed the key distribution problem. This was a technology tailor made for the internet. Customers could send credit card details and send them to retailers on the other side of the planet. It formed the basis of all kinds of modern day communications.

AUTHORS CONTRIBUTION:

The authors suggest an innovation in the age-old conventional cryptographic technique of HILL - CIPHER using the concept of self repetitive matrix. A numerical method has been stated mathematically proved and later implemented ingenerating a random matrix of given periodicity. An investigation was also carried out to find out a definitive correlation between the Eigen value of a periodic matrix and its periodicity. As a result the age old fault in hill cipher technique which surfaces if the inverse of the multiplicative matrix does not exist has been compromised. This method is easier to implement compared to other techniques like self invertible matrix etc. and if the block size and the modular index is suitably chosen that it becomes extremely tough to crack it by brute force as it is not very easy to find multiple inverses of a higher square matrix of higher order.

CHAPTER 2

2.1 What is Cryptography?

Definition: Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication. Cryptography is not the only means of providing information security, but rather one set of techniques.

2.1.1Cryptographic goals:

1. Confidentiality is a service used to keep the content of information from all but those authorized to have it. Secrecy is a term synonymous with confidentiality and privacy. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms which render data unintelligible.
2. Data integrity is a service which addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties. Data manipulation includes such things as insertion, deletion, and substitution.
3. Authentication is a service related to identification. This function applies to both entities and information itself. Two parties entering into a communication should identify each other. Information delivered over a channel should be authenticated as to origin, date of origin, data content, time sent, etc. For these reasons this aspect of cryptography is usually subdivided into two major classes: entity authentication and data origin authentication. Data origin authentication implicitly provides data integrity (for if a message is modified, the source has changed).
4. Non-repudiation is a service which prevents an entity from denying previous commitments or actions. When disputes arise due to an entity denying that certain actions were taken, a means to resolve the situation is necessary. For example, one entity may authorize the purchase of property by another entity and later deny such authorization was granted. A procedure involving a trusted third party is needed to resolve the dispute.

2.1.2 Classification:

Cryptographic systems are generally classified along three independent dimensions:

1. Type of operations used for transforming plaintext to cipher text. All encryption algorithms are based on two general principles. Those are substitution, in which each element in the plain text is mapped into another element and transposition in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost. Most systems referred to as product systems, involved multiple stages of substitution and transposition.
2. The number of keys used: If sender and receiver use the same key, the system is referred to as symmetric, single key or secret key conventional encryption. If the sender and the receiver each uses a different key the system is referred to as asymmetric, two key, or public-key encryption.
3. The way in which the plaintext is processed: A block cipher processes the input on block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along.

2.2 Algorithms and Keys

A cryptographic algorithm, also called a cipher, is the mathematical function used for encryption and decryption. (Generally, there are two related functions: one for encryption and the other for decryption).

2.2.1 Terminology

If the security of an algorithm is based on keeping the way that algorithm works a secret, it is a restricted algorithm. Restricted algorithms have historical interest, but are woefully inadequate by today's standards. A large or changing group of users cannot use them, because every time a user leaves the group everyone else must switch to a different algorithm. If someone accidentally reveals the secret, everyone must change their algorithm.

Even more damning, restricted algorithms allow no quality control or standardization. Every group of users must have their own unique algorithm. Such a group can't use off-the-shelf hardware or software products; an eavesdropper can buy the same product and learn the

algorithm. They have to write their own algorithms and implementations. If no one in the group is a good cryptographer, then they won't know if they have a secure algorithm.

Despite these major drawbacks, restricted algorithms are enormously popular for low-security applications. Users either don't realize or don't care about the security problems inherent in their system.

Modern cryptography solves this problem with a key, denoted by K. This key might be any one of a large number of values. The range of possible values of the key is called the keyspace. Both the encryption and decryption operations use this key (i.e., they are dependent on the key and this fact is denoted by the K subscript), so the functions now become:

$E_k(M) = C$

$D_k(C) = M$

Those functions have the property that (see Figure 1.2):

$D_k(A_k(M)) = M$

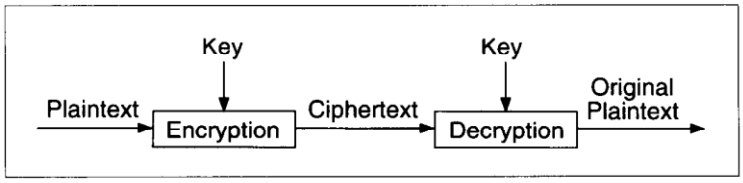
Some algorithms use a different encryption key and decryption key . That is, the encryption key, K_1, is different from the corresponding decryption key, K_2. In this case:

$E_{k1}(M) = C$

$D_{k2}(C) = M$

$D_{k2}(E_{k1}(M)) = M$

All of the security in these algorithms is based in the key (or keys); none is based in the details of the algorithm. This means that the algorithm can be published and analyzed. Products using the algorithm can be mass-produced. It doesn't matter if an eavesdropper knows your algorithm; if she doesn't know your particular key, she can't read your messages.



A cryptosystem is an algorithm, plus all possible plaintexts, ciphertexts, and keys.

2.3 Symmetric Algorithms

There are two general types of key-based algorithms: symmetric and public-key. Symmetric algorithms, sometimes called conventional algorithms, are algorithms where the encryption key can be calculated from the decryption key and vice versa. In most symmetric algorithms, the encryption key and the decryption key are the same. These algorithms, also called secret-key algorithms, single-key algorithms, or one-key algorithms, require that the sender and receiver agree on a key before they can communicate securely. The security of a symmetric algorithm rests in the key; divulging the key means that anyone could encrypt and decrypt messages. As long as the communication needs to remain secret, the key must remain secret.

Encryption and decryption with a symmetric algorithm are denoted by:

$E_k(M) = C$

$D_k(C) = M$

Symmetric algorithms can be divided into two categories. Some operate on the plaintext a single bit (or sometimes byte) at a time; these are called stream algorithms or stream ciphers. Others operate on the plaintext in groups of bits. The groups of bits are called blocks, and the algorithms are called block algorithms or block ciphers. For modern computer algorithms, a typical block size is 64 bits large enough to preclude analysis and small enough to be workable. (Before computers, algorithms generally operated on plaintext one character at a time. You can think of this as a stream algorithm operating on a stream of characters.)

2.4 Public-Key Algorithms

Public-key algorithms (also called asymmetric algorithms) are designed so that the key used for encryption is different from the key used for decryption. Furthermore, the decryption key cannot (at least in any reasonable amount of time) be calculated from the encryption key. The algorithms are called “public-key” because the encryption key can be made public: A complete stranger can use the encryption key to encrypt a message, but only a specific person with the corresponding decryption key can decrypt the message. In these systems, the encryption key is often called the public key, and the decryption key is often called the private key. The private key is sometimes also called the secret key, but to avoid confusion with symmetric algorithms, that tag won’t be used here.

Encryption using public key K is denoted by:

$E_k(M) = C$

Even though the public key and private key are different, decryption with the corresponding private key is denoted by:

$D_k(C) = M$

Sometimes, messages will be encrypted with the private key and decrypted with the public key; this is used in digital signatures. Despite the possible confusion, these operations are denoted by, respectively:

$E_k(M) = C$

$D_k(C) = M$

2.5 Symmetric-key vs. public-key cryptography

Symmetric-key and public-key encryption schemes have various advantages and disadvantages, some of which are common to both. This section highlights a number of these and summarizes features pointed out in previous sections.

2.5.1 (i) Advantages of symmetric-key cryptography

1. Symmetric-key ciphers can be designed to have high rates of data throughput. Some hardware implementations achieve encrypt rates of hundreds of megabytes per second,while software implementations may attain throughput rates in the megabytes per second range.
2. Keys for symmetric-key ciphers are relatively short.
3. Symmetric-key ciphers can be employed as primitives to construct various cryptographic mechanisms including pseudorandom number generators, hash functions, and computationally efficient digital signature schemes, to name just a few.
4. Symmetric-key ciphers can be composed to produce stronger ciphers. Simple transformations which are easy to analyze, but on their own weak, can be used to construct strong product ciphers.
5. Symmetric-key encryption is perceived to have an extensive history, although it must be acknowledged that, notwithstanding the invention of rotor machines earlier, much of the knowledge in this area has been acquired subsequent to the invention of the digital

computer, and, in particular, the design of the Data Encryption Standard in the early 1970s.

2.5.2 (ii) Disadvantages of symmetric-key cryptography

1. In a two-party communication, the key must remain secret at both ends.
2. In a large network, there are many key pairs to be managed. Consequently, effective key management requires the use of an unconditionally trusted TTP.
3. In a two-party communication between entities A and B, sound cryptographic practice dictates that the key be changed frequently and perhaps for each communication session.
4. Digital signature mechanisms arising from symmetric-key encryption typically require either large keys for the public verification function or the use of a TTP.

2.5.3 (iii) Advantages of public-key cryptography

1. Only the private key must be kept secret (authenticity of public keys must, however, be guaranteed).
2. The administration of keys on a network requires the presence of only a functionally trusted TTP as opposed to an unconditionally trusted TTP. Depending on the mode of usage, the TTP might only be required in an "off-line" manner, as opposed to in real time.
3. Depending on the mode of usage, a private key/public key pair may remain unchanged for considerable periods of time, e.g., many sessions (even several years).
4. Many public-key schemes yield relatively efficient digital signature mechanisms. The key used to describe the public verification function is typically much smaller than for the symmetric-key counterpart.
5. In a large network, the number of keys necessary may be considerably smaller than in the symmetric-key scenario.

2.5.4 (IV) Disadvantages of public-key encryption

1. Throughput rates for the most popular public-key encryption methods are several orders of magnitude slower than the best known symmetric-key schemes.
2. Key sizes are typically much larger than those required for symmetric-key encryption , and the size of public-key signatures is larger than that of tags providing data origin authentication from symmetric-key techniques.

3. No public-key scheme has been proven to be secure (the same can be said for block ciphers). The most effective public-key encryption schemes found to date have their security based on the presumed difficulty of a small set of number-theoretic problems.
4. Public-key cryptography does not have as extensive a history as symmetric-key encryption, being discovered only in the mid 1970.

2.6 Digital signatures

A cryptographic primitive which is fundamental in authentication, authorization, and no repudiation is the digital signature. The purpose of a digital signature is to provide a means for an entity to bind its identity to a piece of information. The process of signing entails transforming the message and some secret information held by the entity into a tag called a signature. A generic description follows.

Nomenclature and set-up

1. M is the set of messages which can be signed.
2. S is a set of elements called signatures, possibly binary strings of a fixed length.
3. S_A is a transformation from the message set M to the signature set S, and is called a signing transformation for entity A. The transformation S_A is kept secret by A, and will be used to create signatures for messages from M.
4. V_A is a transformation from the set M x S to the set {true, false}. V_A is called a verification transformation for A's signatures, is publicly known, and is used by other entities to verify signatures created by A.

2.6.1 Signing procedure

Entity A (the signer) creates a signature for a message m E M by doing the following:

1. Compute $s = S_A(m)$.
2. Transmit the pair (m, s). s is called the signature for message m.

2.6.2 Verification procedure

To verify that a signature s on a message m was created by A, an entity B (the verifier) performs the following steps:

1. Obtain the verification function V_A of A.
2. Compute $u = V_A(m, s)$.
3. Accept the signature as having been created by A if u = true, and reject the signature if u = false.

.

2.6.3 Properties required for signing and verification functions

There are several properties which the signing and verification transformations must satisfy.

(a) s is a valid signature of A on message m if and only if VA(m, s) = true.

(b) It is computationally infeasible for any entity other than A to find, for any m E M, an s E S such that $V_A(m, s)$ = true.

No one has yet formally proved that digital signature schemes satisfying (b) exist (although existence is widely believed to be true); however, there are some very good candidates.

2.7 RSA

Soon after Merkle's knapsack algorithm came the first full-fledged public-key algorithm, one that works for encryption and digital signatures: RSA. Of all the public-key algorithms proposed over the years, RSA is by far the easiest to understand and implement. It is also the most popular. Named after the three inventors-Ron Rivest, Adi Shamir, and Leonard Adleman-it has since withstood years of extensive cryptanalysis. Although the cryptanalysis neither proved nor disproved RSA's security, it does suggest a confidence level in the algorithm.

RSA gets its security from the difficulty of factoring large numbers. The public and private keys are functions of a pair of large (100 to 200 digits or even larger) prime numbers. Recovering the plaintext from the public key and the ciphertext is conjectured to be equivalent to factoring the product of the two primes.

To generate the two keys, choose two random large prime numbers, p and q. For maximum security, choose p and q of equal length. Compute the product:

$$n = pq$$

Then randomly choose the encryption key, e, such that e and (p - l)(q - 1) are relatively prime. Finally, use the extended Euclidean algorithm to compute the decryption key, d, such that

$$ed = 1 \bmod (p-1)(q-1)$$

In other words,

$$d = e^{-1} \bmod ((p-1)(q-1))$$

Note that d and n are also relatively prime. The numbers e and n are the public key; the number d is the private key. The two primes, p and q, are no longer needed. They should be discarded, but never revealed.

To encrypt a message m, first divide it into numerical blocks smaller than n (with binary data, choose the largest power of 2 less than n). That is, if both p and q are l00-digit primes, then n will have just under 200 digits and each message block, m, should be just under 200 digits long. (If you need to encrypt a fixed number of blocks, you can pad them with a few zeros on the left to ensure that they will always be less than n.) The encrypted message, c, will be made up of similarly sized message blocks, c_i, of about the same length. The encryption formula is simply

$$c_i = m_i^e \bmod n$$

To decrypt a message, take each encrypted block c_i and compute

$$m_i = C_i^d \bmod n$$

Since

$$c_i^d = (m_i^e)^d = m_i^{ed} = m_i^{k(p-1)(q-1)+1} = m_i m_i^{k(p-1)(q-1)} = m_i * 1 = m_i \text{ ; all (mod n)}$$

the formula recovers the message.

The message could just as easily have been encrypted with d and decrypted with e; the choice is arbitrary. I will spare you the number theory that proves why this works; most current texts on cryptography cover it in detail.

A short example will probably go a long way to making this clearer. If p = 47 and q= 71, then

Public Key:

n product of two primes, p and q (p and q must remain secret)

e relatively prime to (p - l)(q - 1)

Private Key:

d $e^{-1} \bmod ((p-1)(q-1))$

Encrypting:

$c=m^e \bmod n$

Decrypting:

$m=c^d \bmod n$

$n= pq =3337$

The encryption key, e, must have no factors in common with

(p- l)(q- 1)=46 l 70=3220

Choose e (at random) to be 79. In that case

$d = 79^{-1} \bmod 3220 = 1019$

This number was calculated using the extended Euclidean algorithm. Publish e and n, and keep d secret. Discard p and q.

To encrypt the message

m=6882326879666683

first break it into small blocks. Three-digit blocks work nicely in this case. The message is split into six blocks, m_i, in which

$m_1 = 688$

$m_2 = 232$

$m_3 = 687$

$m_4 = 966$

$m_5 = 668$

$m_6 = 003$

The first block is encrypted as

$688^{79} \bmod 3337 = 1570 = c_1$

Performing the same operation on the subsequent blocks generates an encrypted message:

c=1570 2756 2091 2276 2423 158

Decrypting the message requires performing the same exponentiation using the

decryption key of 1019, so

$1570^{1019} \bmod 3337 = 688 = m_1$

The rest of the message can be recovered in this manner.

Speed of RSA

In hardware, RSA is about 1000 times slower than DES. The fastest VLSI hardware implementation for RSA with a 51%bit modulus has a throughput of 64 kilobits per second

[ZSS]. There are also chips that perform 1024bit RSA encryption. Currently chips are being planned that will approach 1 megabit per second using a 512-bit modulus; they will probably be available in 1995. Manufacturers have also implemented RSA in smart cards; these implementations are slower. In software, DES is about 100 times faster than RSA. These numbers may change slightly as technology changes, but RSA will never approach the speed of symmetric algorithms.

2.8 Data Encryption Standard:

Data encryption standard is the most widely used method of data encryption using a secret key. There are 72,000,000,000,000,000 (72 quadrillion) or more possible encryption keys that can be used. For each given message, the key is chosen at random from among this enormous number of keys. Like other private key cryptographic methods, both the sender and the receiver must know and use the same private key. It was developed in the 1970s by the National Bureau of Standards with the help of the National Security Agency. Its purpose is to provide a standard method for sensitive commercial and unclassified data. IBM created the first draft of the algorithm, calling it LUCIFER. DES officially became a standard in November of 1976.

Data encryption algorithm has a 64-bit block size and uses a 56bit key during execution (8 parity bits are stripped of from the full 64-bit key). The DEA can also be used for single user encryption, such as to store files in a hard in encrypted form. NIST re-certifies DES every 5 years. DES has been in world wide use for over 20 years, and due to the fact that it is a defined standard that any system implementing DES can communicate with any other system using is it.

2.8.1 DES Encryption:

DES is a symmetric, block-cipher algorithm with a key length of 64 bits, and the algorithm operates on successive 64 bit blocks of plain text. Due to symmetric, the same key is used for encryption and decryption, and also uses the same algorithm for encryption and decryption.

Initially a transposition is carried out according to a set table (the initial permutation), the 64-bit plaintext block is then split into two 32-bit block, and 16 identical operations called rounds are carried out on each half. The two halves are joined back together , and the reverse of the initial permutation is carried out. The purpose of the first transposition is clear; it does not affect the security of the algorithm, but is through for the purpose of allowing plaintext and

cipher text to be loaded into 8-bit chip in byte-sized pieces. In any round, only one half of the original 64-bit block is operated on. The rounds alternate between the two halves. One round in DES consists of the following:

2.8.1.1 Key Transformation:

The keys reduced from 64-bit to 56-bit by removing every eighth bit, which are sometimes used for error checking. 16 different 48 bit sub-keys are then generated i.e. one for each round. This is achieved by splitting the 56-bit key into the two halves, and then circularly shifting them left by one or two bits, depending on the round .After this, 48 of the bits are selected. Because they are shifted, different groups of key bits are used in each sub key. This process is called compression permutation due to transposition of bits & reduction of the overall size.

2.8.1.2 Expansion Permutation:

Whichever half of the block is being operated on undergoes a permutation after key transformation. In this operation, the expansion & the transposition are achieved simultaneously by allowing the first and fourth bits in each block for bit block to appear twice in the output that is the fourth input bit becomes the fifth and the seventh output bit.

The expansion permutation achieves 3 things. Those are described below:

1) It increases the size of the half block from 32 to 48 bit, the same number of bit as in compressed key subset, which is important as the next operation is to XOR the two together.
2) It produces a long string of data for the substitution operation that subsequently comprises it.
3) Because in the subsequent substitutions on the first & 4th bits appearing in 2 Sboxes, they affect two substitutions. The effect of this is the dependency of the output on the input bits is that the dependency of the output on the input bits increases rapidly.

2.8.1.3 XOR:

XOR operation performed with the appropriate subset key for that round & the resulting 48 block.

2.8.1.4 Substitution:

After XOR operation the next operation is to perform substitution on the expanded block. There are 8 substitution boxes called S-boxes. The first S-Box operates on the first 6 bits of the 48 bit expanded block, the second S-box on the next 6 & so on. Each S-box operates from a table of 4 rows & 16 columns; each entry on a table is a 4 bit number. The 6 bit number the s-box takes as input is used to look up the appropriate entry in the table in the following way. The first and the 6 bits combine to form a two bit number corresponding to a row number, the second and fifth bit combine to form a 4 bit number corresponding to a particular column. The net result of the substitution phase is 8 4bit blocks that are then combined to form a 32 bit block. It is the non-linear relationship of the S-boxes that really provides DES with its security.

2.8.1.5 Permutation:

The 32 bit output of a substitution phase then undergoes a straight forward transposition using a table called P-Box. After all the round has been completed, the two half blocks of 32 bits are recombined to form a 64 bit output. The final permutation is performed on it, and the resulting 64 bit block is the desired DES encrypted cipher text of the input plain text block.

2.8.2 DES Decryption:

If one has the correct key decrypting DES is very easy. The decryption algo is identical to the encryption algo. The only change is to decrypt DES cipher text; the subsets of the keys use in each round are used in reverse, which is the 16th subset is used first.

2.8.3 Security of DES:

DES can no longer be considered a sufficiently secured algorithm if the DES secured message can be broken in minutes by a super computer. Then the rapidly increasing power of computer means, it'll be trivial to break DES in future. An extension of DES called DESX is considered virtually immune to key search.

2.9 International Data Encryption Algorithm:

The international data encryption algorithm is a symmetric block cipher developed by Xuejia Lai & James Massey in the Swiss federal institute of Technology in 1990 and was called the proposed encryption standard PES. In 1991, Lai & Massey strengthened the algorithm against differential crypt analysis and called the result improved PES. The IPES name was changed to International Date Encryption Algorithm in 1992.

IDEA is one of a number of conventional encryption algorithms that have been proposed in recent years to replace DES. In terms of adoption, IDEA is one of the most successful of these proposals. IDEA is best known for its use in PGP (pretty good privacy) in network protocol.

2.9.1 The Algorithm

IDEA algorithm with the key length of 128 bits, a block size of the 64 bits and as with DES, the same algorithm provides encryption and decryption. IDEA consists of 8 rounds using 52 sub keys. Each round uses 6 sub keys with the remaining 4 being used for output transformation. The Sub-keys are created as follows:

1) The 128 bit key is divided into 8 16 bit keys to provide the first 8 sub keys.
2) The bits of the original key are then shifted 25 bits to the left, and then it is again split in 8 sub keys.
3) The shifting & splitting is repeated until all 52 sub keys (SK1-SK52) have been created.

The 64 bit plain text is first split into four blocks. A round then consists of the following steps:

Output block-1 (OB1) = b1*sk1 (multiply 1st sub-block with 1st sub key)
(OB2) = b2+sk2 (add 2nd sub-block with 2nd sub key)
(OB3) = b3+sk3 (add 3rd sub-block with 3rd sub key)
(OB4) = b4*sk4 (multiply 4th sub-block with 4th sub key)
(OB5) = OB1 XOR OB3 (XOR results of 1 & 3)
(OB6) = OB2 XOR OB4
(OB7) = OB5 * SK5
(OB8) = OB6+OB7
(OB9) = OB8*SK6
(OB10) = OB7+OB9
(OB11) = OB1 XOR OB9
(OB12) = OB3 XOR OB9
(OB13) = OB2 XOR OB10
(OB14) = OB4 XOR OB10

The input to the next round is the four sub blocks OB11, OB13, OB12, and OB14 in that order.

After the eighth round, the four final output blocks (F1-F4) are used in a final transformation to produce 4 sub blocks of cipher text c1-c4 that are then rejoined to form final 64 bit block of cipher text.

C1=F1*SK49

C2=F2+SK50

C3=F3+SK51

C4=F4+SK52

Cipher Text= C1 C2 C3 C4

2.9.2 Security provided by IDEA:

IDEA is approximately twice as fast as DES and is also considerably more secure. Using a brute force approach there are 2128 possible keys. If a billion chips that could each test 1 billion keys a second would try and crack an IDEA encrypted message, it would take them 1013 years. Being a fairly new algorithm, it is possible a better attack than brute force will be found, when coupled with more powerful machines in the future may be able to crack a message. However, a long way into future IDEA seems to be a very secure algorithm.

2.10 Blowfish:

Blowfish is a variable-length key block cipher developed by Bruce Schneier. It does not meet all the requirements for a new cryptographic standard discussed above: It is only suitable for applications where the key does not change often, like a communications link or an automatic file encryptor. It is significantly faster than DES when implemented in 32-bit microprocessors with large data caches, such as the Pentium and the Power PC.

2.10.1 Description of the Algorithm:

Blowfish is a variable-length key, 64-bit block cipher. The algorithm consists of two parts; a key-expansion part and a date-encryption part. Key expansion converts a key of at most 448 bits into several sub-key arrays totaling 4168 bytes. Data encryption occurs via a 16-round Feistel network. Each round consists of a key dependent permutation, and a key-and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.

2.10.2 Sub-keys:

Blowfish uses a large number of sub-keys. These keys must be pre-computed before any

data encryption or decryption.

1. The P-array consists of 18 32-bit sub-keys:

   P1, P2…P18

2. There are four 32-bit S-boxes with 256 entries each:

   S1, 0, S1, 1 ……., S1, 255;

   S2, 0 S2, 1… S2, 255;

   S3, 0, S3, 1… S3, 255;

   S4, 0, S4, 1… S4, 255;

The exact method used to calculate these sub-keys will be described later.

2.10.3 Encryption and Decryption:

Blowfish is a Feistel network consisting of 16 rounds. The input is a 64-bit data element, X

Divide X into two 32-bit halves XL, XR

For I = 1 to 16

XL = XL XOR Pi

XR = F (XL) XOR XR

Swap XL and XR

Swap XL and XR (Undo the last swap)

XR = XR XOR P17

XL = XL XOR P18

Recombine XL and XR

Function F:

Divide XL into four eight-bit quarters: a, b, c and d

F (XL) = ((S1,a + S2,b + mob 2 32) XOR S3,C) + S4,d MOB 232 (2.25)

Decryption is exactly the same as encryption, except that P1, P2 … P18 are used in the reverse order.

Implementations of Blowfish that require the fastest speeds should unroll the loop and ensure that all sub keys are stored in cache.

Generating the Sub keys:

The sub keys are calculated using the Blowfish algorithm. The exact method is as

follows:

1. Initialize first the P-array and then the four S-boxes, in order with fixed string.
   This string consists of the hexadecimal digits of fractional part of Pi
   (Less the initial 3). For example:
   P1 = 243f6a88
   P2 = 85a308d3
   P3 = 13198a2e
   P4 = 03707344
2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits. (For every short key, there is a least one equivalent longer key' for example, if A is a 64-bit key, then AA, AAA, etc equivalent keys.)
3. Encrypt the all-zero string with the Blowfish algorithm using the sub keys described in steps (1) and (2).
4. Replace P1 and P2 with the output of step (3).
5. Encrypt the output of step (30 using the Blowfish algorithm with the modified sub-keys.
6. Replace P3 and P4 with the output of step (5).
7. Continue the process, replacing all entries of the P-array, and then all four Sboxes in order, with the output of the continuously changing Blowfish algorithm. In total, 521 iterations are required to generate all required sub-keys. Applications can store the sub-keys rather than derivation process multiple times.

2.11 RC Cipher:

RC stands for *Ron's Code or Rivest Cipher*. These ciphers were designed by Ron Rivest for the RSA Data Security. Different RC Ciphers described briefly below.

2.11.1 RC2:

It was designed as a quick-fix replacement for DES that is more secure. It is a block cipher with a variable key size that has propriety algorithm RC2 is a variable-key length cipher. However, when using the Microsoft base cryptographic provider, the key-length is hard –coded to 40 bits. When using the Microsoft enhanced cryptographic provider, the key length is 128 bits by default and can be in the range of 40 to 128 bit in 8-bit increments.

2.11.2 RC4:

It was developed by Ron Rivest in 1987. It is a variable-key-size stream cipher. The details of the algorithm have not been officially published. The algorithm is extremely easy to describe and program just like RC2, 40 bit RC4 is supported by the Microsoft base Cryptography provider, and the enhanced provider allows keys in the range of 40 to 128 bits in 8-bit increments.

2.11.3 RC5:

RC5 is a block designed for speed. It allows a user defined key length, data block size, and number of encryption rounds. In particularly the key size can be as large as 2,048 bits. RSA Data security is working to have RC5 included in numerous internet standards including 1Psec.

CHAPTER 3

3.1 HILL CIPHER

The core of Hill-cipher is matrix manipulations. It is a multi-letter cipher, developed by the mathematician Lester Hill in 1929. For encryption, algorithm takes m successive plaintext letters and instead of that substitutes m cipher letters. In Hill cipher each character is assigned a numerical value like:

a=0,

b=1,

…..

…..

z=25.

The substitution of cipher text letters in place of plaintext leads to m linear equations. For m=3, the system can be described as follows:

$C1=(K_{11}P_1+K_{12}P_2+K_{13}P_3)MOD26$---------(2.3)

$C1=(K_{21}P_1+K_{22}P_2+K_{23}P_3)MOD26$---------(2.4)

$C1=(K_{31}P_1+K_{32}P_2+K_{33}P_3)MOD26$---------(2.5)

This can be expressed in terms of column vectors and matrices:

C=KP

Where C and P are column vectors of length 3, representing the plaintext and the cipher text and K is a 3*3 matrix, which is the encryption key. All operations are performed mod 26 here. Decryption requires the inverse of matrix K. The inverse K^{-1} of a matrix K is defined by the equation.

$K K^{-1}$= I where I is the Identity matrix.

NOTE: The inverse of a matrix doesn't always exist, but when it does it satisfies the proceeding equation.

K^{-1} is applied to the cipher text, and then the plain text is recovered. In general terms we can write as follows:

For encryption: $C=E_k(P)=K_p$

For decryption: $P=D_k(C) =K^{-1} C= K^{-1}K_p=P$

3.2 Novel Modification to the Algorithm

As we have seen in Hill cipher decryption, it requires the inverse of a matrix. So while one problem arises that is:

Inverse of the matrix doesn't always exist. Then if the matrix is not invertible then encrypted text cannot be decrypted.

In order to overcome this problem author suggests the use of self repetitive matrix. This matrix if multiplied with itself for a given mod value (i.e. mod value of the matrix is taken after every multiplication) will eventually result in an identity matrix after N multiplications. So, after N+ 1 multiplication the matrix will repeat itself.

Hence, it derives its name i.e. self repetitive matrix. It should be non singular square matrix.

Note: Historically, there are already many available methods, like self invertible matrix etc. But this method as suggested by the author is both easy to compute and simpler to implement.

3.3 Modular Arithmetic: A brief Analysis

The analysis presented here for generation of self repetitive matrix is valid for matrix of positive integers that are the residues of modulo arithmetic on a prime number. So in analysis the arithmetic operations presented here are addition, subtraction, Unary operation, Multiplication and division.

The Modulo operator have the following properties:

1. a=b mod p if n| (a-b)
2. (a mod p)=(b mod p)=>a=b mod p
3. a=b mod p=>b=a mod p
4. a=b mod p and b=a mod p=>a=c mod p

The modulo arithmetic have the following properties:

Let z= [0, 1,…..,p-1], the set residues modulo p. If modular arithmetic is performed within the set Z_n, the following equations present the arithmetic identities:

1. Addition: (a+b) mod p= [(a mod p)+(b mod p)]mod p
2. Subtraction: (a-b) mod p= [(a mod p)-(b mod p)]mod p
3. Multiplication: (a*b) mod p= [(a mod p)*(b mod p)]mod p
4. Negation: -a mod p=p-(a mod p)
5. Division: (a/b) mod P=c when a=(c*b) mod p
6. Multiplicative inverse: (a-1) =c if there exists (c*z) mod p=1

Table exhibits the properties of modulo arithmetic:

SL.No.	Property	Expression
1.	Commutative Law	. (w + x) mod p=(x + w) mod p (w*x) mod p= (x*w) mod p
2.	Associative Law	[(w + x)+y] mod p = [w+ (x + y)] mod p
3.	Distributive Law	[w*(x + y)] mod p=[w*x +w*y] mod p [w*(x * y)] mod p=[{w*x mod p}*{(w*y)mod p}] mod p
4.	Identities	(0+a) mod p = a mod p and (1*a) mod p= a mod p
5.	Inverse	For each X belongs to Z_p , there exists y such that (x + y) mod p =0 then y= -x For each X belongs to Z_p, there exists y such that (x*y) mod p=1

3.3.1 Complex modular numbers:

Used while calculating negative or non existent square roots.

Say, for example, for mod 7

sqrt(1)=1;

sqrt(2)=4;

sqrt(4)=2;

But square root of 3, 5, 6 doesn't exist

So sqrt(6)=sqrt(-1)sqrt(-6)=j where j stand for square of -1 in modular arithmetic

Similarly, sqrt(5)=4j ; sqrt(3)=2j

Extensive no of simulation exercises were carried out to understand the properties of modular arithmetic.

3.4 Generation of a self repetitive Matrix A for a Given N:

The initial conditions for the existence of a self repetitive matrix are:

1. The matrix should be square.

2. It should be non-singular.

But trying to find out the value of N (the value where the matrix becomes a identity matrix through the method of brute force may not be the best idea always; because the matrix is of dimension greater than 5*5 and with mod index (i.e.) greater than 91 then the brute force technique might take very long time and N value may be in the range of millions. A normal Pentium 4 machine might hang if asked to do the computations for 15*15 matrixes or more. Hence, it would be comfortable to know the value of N and then generate a random matrix accordingly.

This can be done as follows:

1. First a diagonal matrix A is chosen, and then the values powers of each individual element when they reach unity is calculated and denoted as n1, n2, n3…. Now LCM of these values is taken to given the value of N.
2. Now the next step is generate a random square matrix whose N value is same as the N calculated in the previous step.
3. Pick up any random invertible square matrix B
4. Generate $C=B^{-1}AB$
5. The N value of C is also N

Mathematical proof:

$(B^{-1}AB)^N= (B^{-1})^N*(A)^N*(B)^N$

$A^N=I$ as calculated before as it is a diagonal matrix and N is the LCM of all elements

$(B^{-1}B)*(B^{-1}*B)$…….n times=I

3.5 Another Line of Investigation:

In pursuit of finding a many one relationship of N value with matrices led us towards investigating the correlation of N values with l (lambda) Eigen values. A relationship established like this could easily solve the key distribution problem.

The relationship worked out like this:

NOTE: ALL operations are modular.

Say A is any square matrix and B be the diagonal matrix containing the Eigen values of A. Let P(l) be the characteristic Eigen equation of the A.

Then, l^N-1 will be divisible by P (l). (N is the n value of A) only real l values

3.6 Euler's Theorem:

In number theory, **Euler's theorem** (also known as the **Fermat-Euler theorem** or **Euler's totient theorem**) states that if n is a positive integer and a is coprime to n, then

$$a^{\varphi(n)} \equiv 1 \ (mod) n$$

where $\varphi(n)$ is Euler's totient function and "... ≡ ... (mod n)" denotes congruence modulo n.

The theorem is a generalization of Fermat's little theorem, and is further generalized by Carmichael's theorem.

The theorem may be used to easily reduce large powers modulo n. For example, consider finding the last decimal digit of 7^{222}, i.e. 7^{222} (mod 10). Note that 7 and 10 are co prime, and $\varphi(10) = 4$. So Euler's theorem yields $7^4 \equiv 1$ (mod 10), and we get $7^{222} \equiv 7^{4x55+2} \equiv (7^4)^{55}x7^2 \equiv 1^{55}x7^2 \equiv 49 \equiv 9$ (mod 10).

In general, when reducing a power of a modulo n (where a and n are coprime), one needs to work modulo $\varphi(n)$ in the exponent of a:

if $x \equiv y \pmod{\phi(n)}$, then $a^x \equiv a^y \pmod{n}$.

3.6.1 Proofs

Leonhard Euler published a proof in 1736. Using modern terminology, one may prove the theorem as follows: the numbers a which are relatively prime to n form a group under multiplication mod n, the group of units of the ring $\mathbf{Z}/n\mathbf{Z}$. This group has $\varphi(n)$ elements, and the statement of Euler's theorem follows then from Lagrange's theorem.

Another direct proof: if a is coprime to n, then multiplication by a permutes the residue classes mod n that are coprime to n; in other words (writing R for the set consisting of the $\varphi(n)$ different such classes) the sets $\{ x : x \text{ in } R \}$ and $\{ ax : x \text{ in } R \}$ are equal; therefore, their products are equal. Hence, $P \equiv a^{\varphi(n)}P$ (**mod** n) where P is the first of those products. Since P is coprime to n, it follows that $a^{\varphi(n)} \equiv 1$ (**mod** n).

3.7 Fermat's Theorem:

Fermat's little theorem (not to be confused with Fermat's last theorem) states that if p is a prime number, then for any integer a, $a^p - a$ will be evenly divisible by p. This can be expressed in the notation of modular arithmetic as follows:

$$a^p \equiv a \ (mod\ p)$$

A variant of this theorem is stated in the following form: if p is a prime and a is an integer coprime to p, then $a^{p-1} - 1$ will be evenly divisible by p. In the notation of modular arithmetic:

$$a^{p-1} \equiv 1 \ (mod\ p)$$

Another way to state this is that if p is a prime number and a is any integer that does not have p as a factor, then a raised to the $p-1$ power will leave a remainder of 1 when divided by p.

Fermat's little theorem is the basis for the Fermat primality test.

CHAPTER 4

4.1 Value of N for different Matrices:

Although in the actual Hill cipher method we use matrices of order 15x15 we worked on 3x3 matrices as this is for theoretical purpose, so that calculation will be few and easy. Also the prime number taken as modulo is 13 but in the actual method it will be a very large prime number.

The N value for different eigen equation of a matrix is given by:

1. **Characteristic Equation having three unequal roots:**

   The exponent of characteristic equation $\boldsymbol{x^n - 1 = 0}$ is divisible by the characteristic equation whose roots are a, b, c will be for value of n = LCM(n1, n2, n3). Where n1, n2, n3 are the exponents at which a, b, c become equal to 1 respectively.

   Example: Consider A = $\begin{matrix} \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{7} & \mathbf{10} \\ \mathbf{0} & \mathbf{6} & \mathbf{12} \end{matrix}$

   Roots = 1, 2, 3

   n1 = 1; n2 = 12; n3 = 3

   N = LCM(1, 12, 3) = 12

   $A^{12} = \begin{matrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{matrix}$

2. **Characteristic Equation having two equal roots and one different root:**

   The exponent of characteristic equation $\boldsymbol{x^n - 1 = 0}$ is divisible by the characteristic equation whose roots are a, b will be for value of n = LCM(n1*13, n2). Where n1, n2 are the exponents at which a, b become equal to 1.

   But when it is applied for some matrices whose characteristic polynomial is not a minimum polynomial then the N < LCM(n1*13,n2). Further study of this deviation was not possible due to the time constraints.

   Example: Consider A= $\begin{matrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{7} & \mathbf{1} & \mathbf{5} \\ \mathbf{12} & \mathbf{0} & \mathbf{2} \end{matrix}$

Roots = 1, 1, 2

n1 = 1; n2 = 12

N = LCM(1*13, 12) = LCM(13, 12) = 156

$$A^{156} = \begin{matrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{matrix}$$

$$\text{Consider A} = \begin{matrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{7} & \mathbf{1} & \mathbf{6} \\ \mathbf{12} & \mathbf{0} & \mathbf{2} \end{matrix}$$

Roots = 1, 1, 2

n1 = 1; n2 = 12

N = LCM(1*13, 12) = LCM(13, 12) = 156

But it is found that

$$A^{12} = \begin{matrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{matrix}$$

This is because the minimum polynomial of the above matrix is not equal to the characteristic polynomial.

3. Characteristic Equation having a single root:

The exponent of characteristic equation $\boldsymbol{x^n - 1 = 0}$ is divisible by the characteristic equation whose root is 'a' will be for value of n = LCM(factor(13^2-1), n1). Where n1 is the exponents at which 'a' becomes equal to 1.

$$\text{Consider A} = \begin{matrix} \mathbf{0} & \mathbf{0} & \mathbf{2} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{5} & \mathbf{11} & \mathbf{11} \end{matrix}$$

Roots = 5

n1 = 4

N = LCM(factor of (13^2-1), 4) = 4

$$A^4 = \begin{matrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{matrix}$$

Proof:

$$\boldsymbol{x^2 + ax + b = 0}$$

$$(\boldsymbol{x^2})^{\boldsymbol{84}} = [-(\boldsymbol{ax + b})]^{\boldsymbol{84}}$$

$$= (\boldsymbol{ax + b})^{\boldsymbol{78}}.(\boldsymbol{ax + b})^{\boldsymbol{6}}$$

$$= (\boldsymbol{ax + b})^{\boldsymbol{6}}(\boldsymbol{ax + b})^{\boldsymbol{6}} \quad (\text{ fermat theorem} a^p \equiv a\ (mod\ p))$$

$$= (\boldsymbol{ax + b})^{\boldsymbol{12}}$$

$$= 1 \quad (\text{fermat theorem } a^{p-1} \equiv 1\ (mod\ p))$$

$$\therefore\ \boldsymbol{x^{168} = 1}$$

4. Charecteristic Equation with no roots:

The exponent of characteristic equation $\boldsymbol{x^n - 1 = 0}$ is divisible by the characteristic equation which has no roots is a factor(13^3-1).

$$\text{Consider A} = \begin{matrix} \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{10} & \mathbf{11} \end{matrix}$$

Roots = none

N = factor of (13^3-1))

$$A^{1098} = \begin{matrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{matrix}$$

Proof:

$$= -(\boldsymbol{ax^2 + bx + c})$$

$$(\boldsymbol{x^3})^{\boldsymbol{732}} = [-(\boldsymbol{ax^2 + bx + c})]^{\boldsymbol{732}}$$

$$= (\boldsymbol{ax^2 + bx + c})^{\boldsymbol{13\times 56}}(\boldsymbol{ax^2 + bx + c})^{\boldsymbol{4}}$$

$$= (\boldsymbol{ax^2 + bx + c})^{\boldsymbol{56}}(\boldsymbol{ax^2 + bx + c})^{\boldsymbol{4}}$$

$$= (ax^2 + bx + c)^{13\times4}(ax^2 + bx + c)^4(ax^2 + bx + c)^4$$
$$= (ax^2 + bx + c)^4 (ax^2 + bx + c)^4 (ax^2 + bx + c)^4$$
$$= (ax^2 + bx + c)^{12}$$
$$= 1$$

4.2 IMPLEMENTATION OF PROGRAM IN C:

To find N value for characteristic equation with 3 unequal roots:

```
#include <iostream>
#include<iomanip>
using namespace std;
int main()
{

    int a1,b1,a2,b2,a3,b3,c1,c2,c3;
    int A[3][3];
    int B[3][3]={{1,0,0},{0,1,0},{0,0,1}};
    int C[3][3]={{1,0,0},{0,1,0},{0,0,1}};
    int a,b,c;
    int l,m,n;
    int d0,d1,d2;
    int p;
    for(a=2;a<13;a++)
    {
    for(b=1;b<a;b++)
    {
```

```
c=b;
cout<<a<<"\t"<<b<<"\t"<<c<<"\t";
l= 13-(a+b+c)%13;
m= (a*b+b*c+c*a)%13;
n= 13-(a*b*c)%13;
for(a1=0;a1<13;a1++)
{
for(b2=0;b2<13;b2++)
{
for(a2=0;a2<13;a2++)
{
for(b1=0;b1<13;b1++)
{
for(a3=0;a3<13;a3++)
{
for(c1=0;c1<13;c1++)
{
for(b3=0;b3<13;b3++)
{
for(c2=0;c2<13;c2++)
{
for(c3=0;c3<13;c3++)
{
d2=13-(a1+b2+c3)%13;
d1=(a1*c3+b2*c3+a1*b2-a2*b1-a3*c1-b3*c2)%13;
```

```
d0=13-(a1*b2*c3+a2*b3*c1-a2*b1*c3-a3*b2*c1+a3*b1*c2-a1*b3*c2)%13;
if(d2==l&&d1==m&&d0==n)
{
goto calc;
}
}
}
}
}
}
}
}
}
calc:
        A[0][0]=a1;
        A[0][1]=a2;
        A[0][2]=a3;
        A[1][0]=b1;
        A[1][1]=b2;
        A[1][2]=b3;
        A[2][0]=c1;
        A[2][1]=c2;
        A[2][2]=c3;
        p=1;
```

```
while(1)
{
C[0][0]=(B[0][0]*A[0][0]+B[0][1]*A[1][0]+B[0][2]*A[2][0])%13;
C[0][1]=(B[0][0]*A[0][1]+B[0][1]*A[1][1]+B[0][2]*A[2][1])%13;
C[0][2]=(B[0][0]*A[0][2]+B[0][1]*A[1][2]+B[0][2]*A[2][2])%13;
C[1][0]=(B[1][0]*A[0][0]+B[1][1]*A[1][0]+B[1][2]*A[2][0])%13;
C[1][1]=(B[1][0]*A[0][1]+B[1][1]*A[1][1]+B[1][2]*A[2][1])%13;
C[1][2]=(B[1][0]*A[0][2]+B[1][1]*A[1][2]+B[1][2]*A[2][2])%13;
C[2][0]=(B[2][0]*A[0][0]+B[2][1]*A[1][0]+B[2][2]*A[2][0])%13;
C[2][1]=(B[2][0]*A[0][1]+B[2][1]*A[1][1]+B[2][2]*A[2][1])%13;
C[2][2]=(B[2][0]*A[0][2]+B[2][1]*A[1][2]+B[2][2]*A[2][2])%13;
B[0][0]=C[0][0];
B[0][1]=C[0][1];
B[0][2]=C[0][2];
B[1][0]=C[1][0];
B[1][1]=C[1][1];
B[1][2]=C[1][2];
B[2][0]=C[2][0];
B[2][1]=C[2][1];
B[2][2]=C[2][2];
if(B[0][0]==1&&B[0][1]==0&&B[1][0]==0&&B[1][1]==1&&B[0][2]==0&&B[1][2]==0&&
B[2][0]==0&&B[2][1]==0&&B[2][2]==1)
{
    break;
}
```

```
                p++;
                }
                cout<<"\t"<<p<<"\n";

                }
        }
}
```

4.3 CHECK LIST program:

```
#include <iostream>
#include<iomanip>
using namespace std;
int main()
{
int a,b;
int n;
for(a=2;a<13;a++)
        {
                cout<<a<<"\t";
                n=1;
                b=1;
                while(1)
                {
                        b=(b*a)%13;
```

```
            if(b==1)
                {
                 break;
                }
            n++;
        }
        cout<<n<<endl;
    }
}
```

Output:

No.	N value
2	12
3	3
4	6
5	4
6	12
7	12
8	4
9	3
10	6
11	12
12	2

CHAPTER 5

CONCLUSION:

The HILL cipher technique using a novel method of self repetitive matrix was successfully implemented. The numerical method suggested to find N value of a matrix was successfully tested and used in the implementation. The investigation also opened the correlation between Eigen values and N value of a matrix. The value of N for a given matrix can easily be found using the formulas derived.

BIBLIOGRAPHY:

[1] W.Stallings; "Cryptography and Network Security" 2nd Edition, Prentice Hall,1999

[2] Bruce Schneier: Applied Cryptography, 2nd edition, John Wiley & Sons, 1996

[3] Piper,F "Encryption". Security and Detection, Ecos 97. European Conference;

[4] Abrams, M., and Podell, H. "Cryptography" Potentials, IEEE Page No 36-38.

Issue: 1, Volume: 20, Feb-Mar, 2001

[5] Eskiciogiu,A. Litwin,L " Cryptography and Network Security" LOS Alamitos, CA: IEEE computer society press,1987

[6] Garfinkel, S.L; "Public Key Cryptography" , Computer, IEEE, Volume: 29, Issue:6, June 1996.

[7] W.Diffie; M.E.Hell man, "New Directions in Cryptography" IEEE Transactions Information Theory, Nov, pp 644-654

[8] E.Biham and A.Shmir; "Differential C for Cryptanalysis of the Encryption Standard"; Springer- Verilag, 1993

[9] Bidjos,J; "Threats to Private and Public Key Protection" , Compcon Spring '91. Digest of Papers, 25 Feb-1 March 1991

[10] V. Miller; "Uses of Elliptic Curves in Cryptography. In advances in Cryptography, Springer Verlag Crypto 95.